Clock Synchronization in Android Wi-Fi Direct Network

Bachelor of Science Thesis in the Programme Software Engineering&Management

FENGYUAN BAI

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, August 2014

Clock Synchronization in Android Wi-Fi Direct Network

Fengyuan Bai

Examiner: Morgan Ericsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Cover picture taken from: http://www.munasi.co.za/

Department of Computer Science and Engineering
Göteborg, Sweden August 2014

Abstract

Wi-Fi Direct Network is a new type of network and smart-phone devices may utilize this type of network to exchange information, such kind of network is widely replacing Bluetooth as the main communication method among smart-phones. Many new applications and games are developed based on this kind of network, in this thesis, the author explore the clock synchronization problem in Android Wi-Fi Direct network and design a prototype to synchronize the clock time of the Android devices. The prototype considers power overhead and deploys an efficient algorithm to balance the power consumption; meanwhile, the prototype considers multi-hop communication and allows the solution very high scalability.

Categories and Subject Descriptors: Computer communication network, Android, Wi-Fi Direct Network

1. Introduction

Clock synchronization is an old topic and has already been extensively studied in traditional centralized and distributed system. In a nutshell, it drives nodes in a network agrees to a common clock time. This is very important because in many centralized or distributed systems, it does not allow hosts inside the systems have much deviation of clock time, such as in some battle field or scientific exploration, mesh network and other opportunity network. [8] In those conditions, nodes inside the network require a synchronized clock time with each other. Therefore, it is extractive to research on the clock synchronization problem. In recent years, smart-phone is wide spread and such kind of information systems are widely deployed in smart-phone platforms. Android, IOS, Windows phone are the three most popular smart-phone platforms. In this thesis, I will research on the clock synchronization problem based on Android platform and finally comes a solution prototype.

Clocks have deviation due to four factors: frequency accuracy, frequency stability, time accuracy and time stability. The frequency of a clock is how well it realizes the length of the second. This parameter may vary due to hardware and environment interference. The time accuracy means how well a clock agrees to the Coordinated Universal Time (UTC). Time accuracy need to be considered consistent when hosts are spatially distributed. Two other fundamental concepts are synchronization and syntonization. Synchronization means times of hosts in a network are synchronized using some specific methods after taking delay or other interference parameter into account. Syntonization means the adjustment of the electronic circuit to enable the hosts may have the same frequency.

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, August 2014

Generally, clock synchronization refers using some synchronization algorithm to reset the clock time without changing the frequency. The clock devices belonging to the systems may differ after some amount of time because the clock drift are of different rate of local oscillators and can be influenced by the environment. Even if at the beginning devices are of the same rate, after a period of time, the clock time may generate a time error again. Usually, the referring influence can be abstracted into two factors: the clock drift and uncertainty of the delay when the messages exchanged travelling along the distributed network. In a network with low frequency of communication, the clock drift is more influent while if there is frequent communication in the systems, the uncertainty of the delay should be considered more.

The synchronization algorithm usually considered compensate the time error which is generated due to both the clock drift and the uncertainty of the delay that messages carrying synchronization information from sender to receiver. Usually, a clock time outside the network is referenced as external time while the clock time encapsulated in the exchanged messages are referenced as internal time. The external synchronization requires the nodes inside the network exchanges clock time continuously respect to the external system time. The internal synchronization will share the time only among the nodes inside the network. The synchronization algorithm can be deployed in the connected network and help nodes inside to synchronize the clock time.

Network Time Protocol (NTP) is the most dominant protocol for time synchronization (IETF RFC1305, 1992) [1] in packet switched, wired network. It was developed in 1985 by the University of Delaware and based on UDP protocol. NTP is able to achieve very high accuracy (few milliseconds even us) by implementation of the Marzullo algorithm (Marzullo, 1984). The accuracy can be in few milliseconds or even in as few as microseconds. NTP protocol organize nodes inside the network into multiple levels called stratum and allocate a number to each node for identification. There is a number 0 belongs to the server at the top level. Other nodes may query their parent node for the time and the parent nodes will query their parent node recursively until the number 0 node. NTP uses client-server synchronization model to implement the transaction. The transaction is initiated by the client and the client will send a packet storing its timestamp to the server. When the server receives this packet, it will gives response packet back to the client. When the client receives the reply, it can estimate the travelling time of the packet and calculate the propagation time (about the half of packet travelling time). Then it can set its time using the server's time (encapsulated in the reply packet) minus propagation time. This value will be validated several times using checking packets before it is applied to the client. These checks will be implemented several rounds and finally allow the client agrees to a valid clock time. NTP is a successful synchronization solution for packet-switched wired network. However, it does not perform well in wireless network.

Wireless network is increasingly popular than before thanks to its high flexibility and decreasing of cost. Wireless network has its own features and is different with wired network. One of the biggest differences is that wireless network is often composed of a set of wireless devices and such devices are often required to be of low cost, low power, smart with high computation power. The high power consumption for the message exchanging makes traditional clock synchronization protocol such as NTP not that effective. There should be a trade-off between power assumption and accuracy for the synchronization algorithm.

Smart phone devices are the most popular devices which can be connected by different type of wireless network. A smart phone can be connected to wireless network through its Wi-Fi connector or mobile network connector. In such network, a smart-phone device is only working as a client and interacts with a Wi-Fi router or a mobile network server. It cannot change the network or interact with other smart-phone directly. In such network, clock synchronization is also possible to be achieved through traditional synchronization methods such as NTP. As the technology is developed, Wi-Fi technology is improved and the smart-phone is becoming more powerful than before. In the most popular embedded systems such as IOS, android, smart-phones can be connected with each other directly and establish a new, Wi-Fi direct network. In such network, smart-phones are not working only as the client and instead, some one of them can service as a server and provide routing service or data to other smart-phones. Such network is of high convenience and may be used to develop new functions such as connecting a smart phone with printers, laptops, cameras nearby. Or some new kinds of products like wireless mouse, keyboards are also developed. Such kind of network has its own features:

1) The server node requires more power for exchanging data.
2) The network topology is easily to be changed when some nodes join or quit the Wi-Fi direct network.

Therefore, to synchronize clock time in such network is to some level different with traditional wired or wireless network and becoming a new engineering problem. For android operating system, the Wi-Fi direct network is available since version 4.0. Compared with other standards (Bluetooth, etc.) for android devices connected with each other, Wi-Fi direct is much faster and of longer transmission distance. This feature makes it more attractive for application and games developers.

In this thesis, I will make an analysis of several clock synchronization protocols and then based on the selected protocol to design and implement a clock synchronization solution on the android Wi-Fi direct modules. This solution should be able to synchronize the clock time of android devices in a local Wi-Fi direct network. After applying the clock synchronization process, android devices connected in the Wi-Fi direct network should achieve the same clock time with a small deviation. Such kind of solution can be used directly to synchronize the clock time when a public access point is not available and allow two or more android devices synchronize their clock

time. Moreover, such kind of services can be based other services or applications, or games.

Following contents will be organized as following: In section 2, I will research on the background for the solution including the algorithms suitable for the solution and the multiple communication mechanism provided by Android. In section 3 and 4, it will introduce the design and implementation of the solution. In section 5, it will focus on the performance measurement for proposed solution. Finally, it will give the discussion and conclusion in section 6.

2. Background

This section introduces the theoretical background. It consists of the synchronization algorithms and the Android communication mechanisms. The basic idea is to compare potential alternative algorithms and try to find a suitable one for design and implementation.

Algorithms
In the first section, it is explained why traditional protocols are not suitable in a wireless network. This is because the nodes in a wireless network usually have limited power and storage. Traditional protocols contain a large amount synchronization messages which in turn generating big amount of overhead for power and storage overhead. Also, topology of wireless network may change easily, which leads to the estimation of the end-to-end delay unpredictable and unstable. For all of these reasons, many new algorithms for wireless network are proposed in recent years.

Timestamp Synchronization (TSS) (Kay Römer 2001) [2]
This is an on-demand, internal synchronization algorithm. The scheme considers transforming timestamp between participating nodes and calculates the upper and lower bounds of the clock time for node on the next hop transmission path. This algorithm measures the real-time when the synchronization message is exchanged between sender and receiver and uses these real-time values to calculate the differences of the clock time. It based on the measure values to calculate the upper and lower bound of the message delay and based the message delay, and it calculates the upper and lower bound of the clock time for each node. Moreover, it also takes the maximum clock drift for each node into consideration and compensates values for this parameter. The advantage for this algorithm is that it has a low resource and message overhead and therefore it is well suited for resource limited distributed sensor network.

Reference Broadcast Synchronization (RBS) (Jeremy Elson et al, 2002) [3]

In this scheme nodes send reference beacons to their neighbors using physical layer broadcasts. In the beacons there are no timestamp included; instead, receivers reply to the senders with the receiving time with their local time. And afterwards, receivers will exchange this receiving time with other receivers. Each node may observe the receiving time of all other nodes and based on the observation, each node can calculate the offset of to the average receiving time. After the offset is calculated, this scheme uses an algorithm to calculate the clock skew. The network is clustered in case of multi-hop system. In each cluster, there is its own beacon node and these beacon nodes may generate another level synchronization using RBS algorithms. This algorithm can achieve much higher accuracy with low power overhead. However, the disadvantage is that it can be deployed in a simple physical layer broadcast domain and also in this domain there should be a limit for the number of the nodes. Because each nodes have to maintain a timescale for all the nodes nearby and as the set grows, the likelihood increases some of the nodes may be poorly synchronized.

Timing Synchronization for Sensor Network (TPSN) (Ganeriwal et al, 2003) [4] This protocol is specialized in ad-hoc sensor network and can be extended to use in the wireless network. This protocol consists of two-step work. The first step is to generate a hierarchical structure in the network covering related nodes and eventually all nodes in the network structure are synchronized their timestamp with the reference node. In the first step, a node that acts as the gateway between the external clock time and the internal network can be selected as the root which can be assigned to level 0. The root node will broadcast messages to its neighbors and the receivers will get a level greater than the sender. Every node will neglects the new broadcasted messages after it sets its own level. After this phase, every node gets a level value and be connected. In the second phase, synchronization is performed along the edge of the hierarchical structure established earlier using a classical sender-receiver synchronization [5] to implement handshake. During the handshake, clock drifts and communication delay will be considered and sender node and received node may be synchronized by series of message exchanging. Eventually, every node is synchronized to the root node. A timeout mechanism is also applied in this phase to prevent packets collision.

Network Time Protocol (NTP) (RFC1305, 1992) and Simple Network Time Protocol (SNTP) [6] Network Time Protocol (NTP) is the diffused communication protocol. It was developed 1985 and based on UDP transport protocol. NTP extends the use of Marzullo's algorithm (1984) to select time server and applies mechanisms to reduce the jitter generated by the communication delay. NTP can achieve good performance to milliseconds in public internet network and even better performance to less than 1 millisecond in local network. NTP is structure all the participating nodes into a hierarchical tree. Each node in this tree is referenced as "stratum" with a corresponding number. The stratum indicates its distance to the time server. Each node queries its parent stratum or neighbors to get its own stratum. Afterwards, NTP uses a client-server model to synchronize the time. The synchronization is initiated by

clients. The client stores the transmitted packet and the local timestamp. The server that receives the packets stores the received timestamp and reply acknowledgement to the client. After the client receives the acknowledged packet, it can calculate the transmission time delay. This value would be validated through several rounds of "sanity checks" before being applied to the applications. SNTP is a simplified version of NTP. Simple Network Time Protocol Version 4 (SNTPv4) was issued in 2010 and it can deploy when ultimate performance of a full NTP implementation based on RFC 1305 is neither needed or justifies. It still deploys the same algorithm using in NTP but without stores the states. SNTP can be applied to simple devices without causing much power consumption.

Android Wireless Environment

Android is a diffused smart-phone platform. Android devices may communicate with base station through radio or hot spot by Wireless-Fidelity (Wi-Fi) connection. It may use NTP protocol to access the time server on the Internet to synchronize the local time with it. However, since android version 4.0, Wi-Fi-Direct is deployed into this platform and this new feature allows android devices advertise itself as a combination of software access point and peer. This means multiple android devices can be grouped together and generate a peer-to-peer network. This peer-to-peer network can be utilized to develop interesting and attractive applications. To develop an application service to synchronize the time on this local network, it is noted to understand how Wi-Fi-direct works in this situation.

Wi-Fi Direct allows Android 4.0 (API level 14) or later devices with appropriate hardware to be connected without using an immediate access point [7]. The APIs consists of the following main parts:

- Methods that allow android devices to discover request and connect to peers defined in WifiP2PManager class.
- Listeners that allow android devices to be noticed of the success or failure of WifiP2PManager method calls
- Intents that notify android devices of specific events detected by the Wi-Fi Direct framework, such as a dropped connection or a newly discovered peer.

In this thesis, these APIs can be used to develop the clock synchronization services in the Android WiFi-direct network.

3. Design

In this section, it will introduce a prototype for the clock synchronization in the local network constructed by android devices through WiFi-direct connection. The design will consist of two parts:

1) To generate the connected network.
2) To synchronize the clock time.

For the first part, it is possible to generate a centralized network or a distributed network. In a centralized network, it is possible to select one server node to create a time server and listen to other nodes to connect to it. Each client node may send a clock time synchronization request to the server node and wait for the response from the server node. The server node can maintain a table to manage all the client nodes. The advantage of this design is that it is quite simple and easy. The server node is responsible for generating a unique clock time (using its local time) and allowing all other nodes synchronized with this it. The disadvantage is that this structure is difficult to scale, because to exchange handshake messages are quite power consuming and it will generate a big power overhead for the server node. Therefore, it is better to consider a distributed network. Each node may maintain only several neighbors and all the nodes construct a connected tree structure. In this situation, each node will synchronize the clock time with its parent node. Each node will only have several child nodes and therefore each node does not need to spend much power for message exchanging. This idea is the same as the algorithm of Timing Synchronization for Sensor Network (TPSN) as mentioned in section 2. Therefore, this algorithm is selected for the solution. This algorithm includes two-phase work: Level Discovery Phase and Synchronize Phase, which can be mapped to above two parts respectively. In following, it will explain the details for this algorithm.

Level Discovery Phase

Initially, there will be a root node assigned a level 0 value and it initiates the level discover phase work by broadcasting a *level discovery packet*. This packet contains the identification and level information of the sender. The immediate neighbors of the root node may assign themselves a level value after receiving the level discovery packet. This level value will always be one larger than the level value of the sender, i.e. if the sender is the root node and the node receives the level discovery packet from the root node will assign itself a level 1. After that, the node may broadcast the level discovery packet itself to its neighbors and this procedure continues. Nodes which are already assigned a level value will neglect future level discovery packet and this may prevent the flooding congestion happens.

Synchronize Phase

In this phase, pair wise synchronization is performed on the connected network established in above. Classical sender-receiver synchronization handshake [5] can be applied here. It can be shown in Figure 1. As is demonstrated, T1, T4 are the clock time measured by local clock time of the sender while T2 and T3 represent the local clock time measured by the receiver. The sender node first sends the *synchronization pulse packet* to the receiver at T1. This packet contains the level value of the sender and time stamp T1. The receiver node receives this packet at T2, where T2 is equal to $T1+\Delta+d$. Δ represents the clock drift between sender and receiver node and d represents the communication delay. The receiver node may send an *acknowledgement packet* at T3 to the sender node. This acknowledgement packet will contain the level value of the receiver node and time stamp T1, T2 and T3. The sender

node receives this packet at T4. Assume the communication delay and clock drift do not change. The sender node may calculate the clock drift Δ and communication delay *d* by:

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2}; \quad d = \frac{(T2 - T1) + (T4 - T3)}{2}; \quad (1)$$

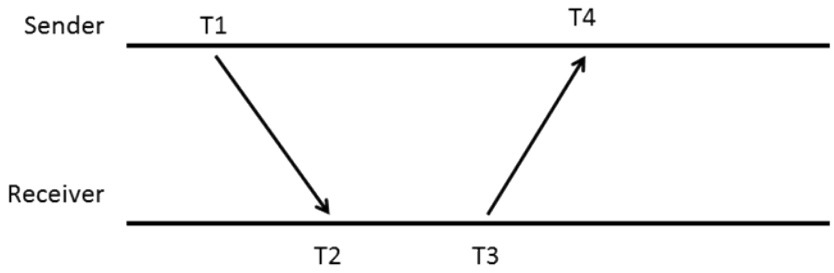


Figure 1: two-way sender-receiver synchronization handshake

The sender node can corrects its local clock time to compensate the communication delay and clock time according to the calculation and synchronize its clock time with the receiver.

This two-way sender-receiver synchronization first starts between the root node and the level 1 node. The root node broadcasts the *time synchronization packet*. On receiving this synchronization packet, the level-1 node may start the synchronization with root node. After level-1 node are synchronized with the root node. It will broadcast the *time synchronization packet* again the level-2 nodes can be synchronized with level-1 nodes. During this procedure, when a node finishes synchronizing with its parent node, it waits for a random time before broadcasts messages to its child nodes. This can make sure the result not influenced by the contention in the medium access. This process is carried out until all nodes are synchronized with the root node.

Apply TPSN to Android WiFi-Direct Network

In TPSN, all nodes can be connected automatically however this cannot be realized for Android devices, because Android device has a protection mechanism for its users. Every time if there is one device that wants to be connected with the other device, it requires to be accepted by the other device. The network is considered to be designed as following:

i) Level 0 node sets up a group with limited size and listens to the requests from other nodes. This group information including the level value can be broadcasted to nearby nodes automatically and detected by other nodes.

ii) Level 1 nodes tries to detect the Level 0 node nearby and sends join request to the group, if the join request is accepted by the Level 0 node, the Level 1 node may sets its level value 1 and get connected with level 0 node. After that, level 1 node sets up its own group with the same group name but change the level value to 1. The group information with new level value can be broadcasted to nearby nodes and detected by other nodes.

iii) Level 2 nodes can connect to the network similarly as level 1 node and this procedure can continue until all the nodes are connected to the network.

After the network is established in above steps, it is possible to start synchronization the clock time. The clock synchronization is always initiated from the high level node to the low level nodes, i.e. Level 1 nodes can synchronize its time with level 0 nodes and level 2 nodes can synchronize its clock time with level 1 nodes.

i) High level node sends synchronization pulse packet to low level nodes, in this synchronization pulse packet, the identification information of the high level node and the sending time T1 is included

ii) When the low level node receives the synchronization pulse packet from high level nodes, there are two possibilities:
   a) The low level node is the root node or it has already synchronized its clock time with root node, then it will record the receiving time of the synchronization pulse packet as T2 and prepare the acknowledgement packet as is done in the TPSN algorithm. The acknowledgement packet will contain the identification of the low level node and all clock time information including the T1, T2 and the sending time of the acknowledgement packet as T3.
   b) The low level node is not the root node and it has not yet synchronized its time with root node, the low level node will suspend the synchronization request and synchronize its time with its parent node as is worked like step i). After the low level node has synchronized with its parent node it may start the step ii.a)

iii) When the high level node receives the acknowledgement packet from the low level node, it first records the receiving time of this packet as T4. Then it can calculate the clock drift and communication delay using the local clock time T1, T2, T3 and T4. The high level node finally corrects its local clock time to compensate the clock drift and communication delay according to the calculation and finish the clock synchronization.

4. Implementation

In this section, it will introduce how the Time Synchronization Protocol for Sensor Network (TPSN) is implemented under Android WiFi-Direct environment.

Integrated Developing Environment (IDE)

In order to develop an Android application or services, it requires configuring the integrated developing environment first. Eclipse integrated with Android SDK is the commonly used IDE for all Android application. I choose the Android SDK version 4.2.2 as the target SDK version which is the latest version when I start the development work. It is noted to mention that only version later than Android SDK

v4.0 can be applied because some key modules are only provided after version Android SDK v4.0.

New Package for Android Peer-to-Peer connection
From Android 4.0, Android support for peer-to-peer (P2P) connections between and other device types without a hotspot or Internet connection. The primary class is **WifiP2PManager**, which can be used to initiate the application for P2P connection, discover nearby devices, and start connection and other more activities.

Construct the connected network
WiFi-Direct service is running as the background service in an Android system and it can be acquired by the system method **getSystemService**(Context.WIFI_P2P_SERVICE). This method may return an instance of the **WiFiP2PManager** type. After that it is possible to use this instance to call the **initialize()** method of **WiFiP2PManager** to initialize this service and then other p2p operation is allowed used. The root node can create a group by calling **createGroup** method; this method may generate an access point waiting for other devices' connection. It is possible to attach the Level information in the parameters of this method and the information can be discovered when requested by other devices. The group information is stored and can be fetched by calling **requestGroupInfo** method with corresponding **WiFiP2PManager** channel and listener information as parameters.
When other devices want to connect to an open group, they first initiate **discoveryPeers** method which may help to do initiation. The calling may stay active until device starts connecting to a peer, forms a p2p group or there is a calling of **stopPeerDiscovery** to stop this operation. After discovering the neighboring peers, one device may call **connect** method to initiate a connecting request to the device creating that group. The group owner has already registered a call back function and when other device call **connect** method, it may trigger call back function to receive this event. Then group owner may create a connection with corresponding device.
As presented in design section, the first node to create the group is given a level 0 value and this level information would be attached in the group information and this value will be acquired by the connected device. The connected device will increase 1 based on this level value and create another group. This operation continues until all the devices are connected in the network.

Synchronize the time
After the connection is set up, it allows using socket to exchange the communication packet. The clock time information for the sender and receiver can be integrated in an xml file which can be composited and de-composited manually. The information of the clock synchronization packet can be designed as following. .

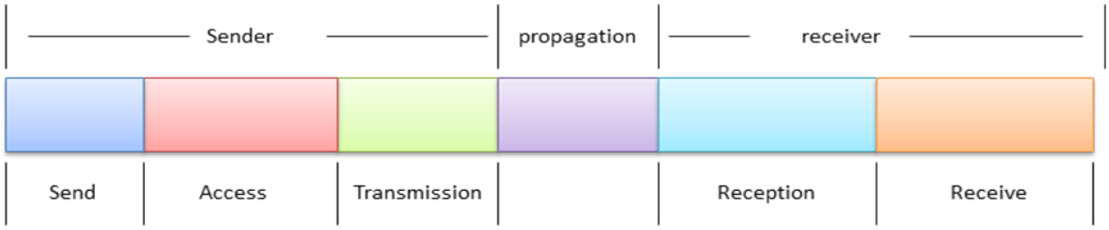


Figure 2: De-composition of the packet in TPSN

Send time: This is a time stamp which represents constructing the packet at the application layer. After the packet is generated in the application layer, it will be passed to MAC layer from the application layer.
Access time: This is the time stamp which represents the time the packet is waiting to access the communication channel. This factor is the most critical factor contributing to the packet delay.
Transmission time: the transmission time is usually fixed and can be estimated using packet size and radio speed.
Propagation time: This is actual time taken by the packet spent on the wireless link from sender to receiver.
Reception time: This refers to the packet receiving by bits and passing them to MAC layer.
Receive time: The packet bits in the MAC layer will be finally passed to the application layer again and decoded by the receiver.
In reality, this model is simplified and it assumes the access time completely overshadows other delays. The **Send** component is used to record the local time of the sender node and **Receive** component is used to record the local time when the receiver node receives the packet. Besides, other components are used to identify the message and the type of the message type: if it is a synchronization request or it is an acknowledgement from the receiver. Then the actual package can be as following.

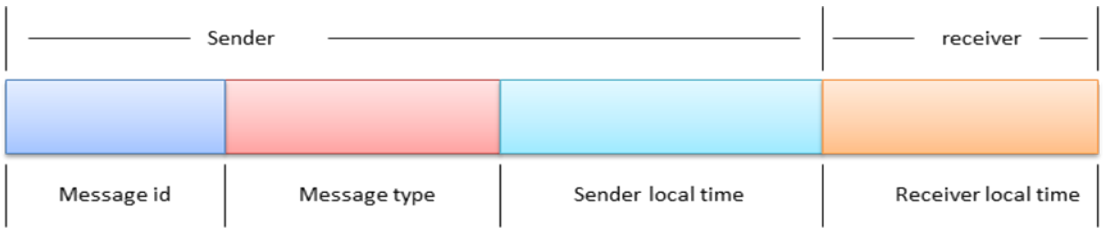


Figure 3: De-composition of the real packet

Needs for resynchronization
In [3], it introduces that in the sensor network, the motes may lose some time in every second, therefore, it requires re-synchronizing the clock drift in every several minutes. In this thesis, it is the android system control the clock time after synchronization. However, Wi-Fi direct network is not stable and the transmission time may vary according to the network quality. There is also condition that the synchronization is not successful and at which moment, the client requires to re-submit the synchronizing request to its high level node. Therefore, in this thesis I change the re-

synchronization method and consider re-synchronize the clock drift manually. When the client submit a re-synchronization request to its higher level node, the prior request is considered invalid and the higher level node may re-execute the synchronization algorithm step by step.

5. Results

In this section, I will introduce how the results are collected and analyzed. The results collection follows [4] and meanwhile the model is simplified because this thesis focus on providing an prototype model and create a software product instead of research on the algorithms. Some measured parameters in the algorithm are not covered in this work and some of the delay sources are combined without separating them. In following, I will introduce how the results are collected and analyzed according to the algorithm.

Error Analysis

I follow [4] to collect parameters and calculate the error. In Figure 2, It shows how the two way connection are established and introduces the related parameters. Herein, I use lowercase letter to represent the real-time clock. It can be easily derived following equations:

$$\mathrm{T2} = \mathrm{T1} + S_A + P_{A\to B} + R_B \quad (2)$$

$$\mathrm{T2} = \mathrm{T1} + S_A + P_{A\to B} + R_B + D_{T1}^{A\to B} \quad (3)$$

Here T1 and T2 represent the measured time of the local clock of sender node and receiver node respectively. S_A, $P_{A\to B}$, R_B refers to the time taken to send the packet (send time + access time + transmission time), propagation time between sender and receiver node and time taken to receive the packet. $D_{T1}^{A\to B}$ refers to the clock drift between sender and receiver node at time T1. The receiver node then sends a reply at T3, which is received by sender node at T4. Using a similar equation:

$$\mathrm{T4} = \mathrm{T3} + S_B + P_{B\to A} + R_A - D_{T4}^{A\to B} \quad (4)$$

It may break $D_{T1}^{A\to B}$ into two components and get following equation:

$$D_{T1}^{A\to B} = D_{T4}^{A\to B} + RD_{T1\to T4}^{A\to B} \quad (5)$$

Here $RD_{T1\to T4}^{A\to B}$ represents the relative clock drift from sender node to receiver node during T1 to T4. If we subtract equation (4) and (3) and using equation (1) and (5), it may get:

$$2*\Delta = S^{uc} + P^{uc} + R^{uc} + RD_{T1\to T4}^{A\to B} + (2*D_{T4}^{A\to B}) \quad (6)$$

S^{uc}, P^{uc}, R^{uc} represents the uncertainty at sender, receiver and in propagation time respectively, and they can be calculated by following equations:

$$S^{uc} = S_A - S_B \quad (7)$$

$$R^{uc} = R_A - R_B \quad (8)$$

$$P^{uc} = P_{A\to B} - P_{B\to A} \quad (9)$$

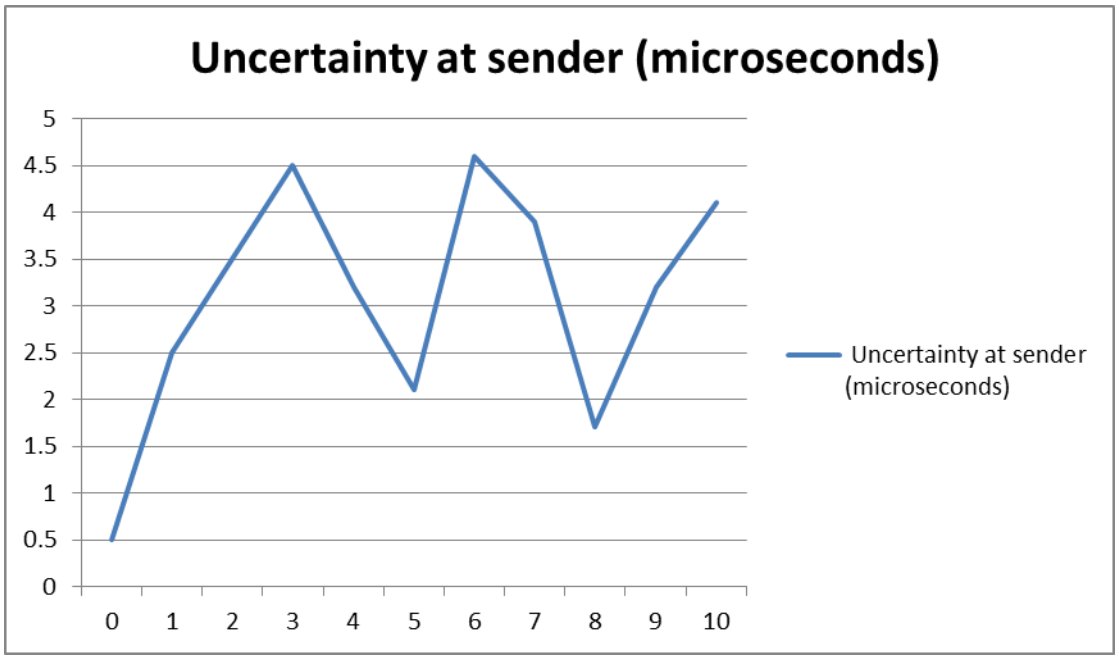
It is noted to mention that we correct the time at T4 and therefore to calculate $D_{T4}^{A\to B}$ is our aim, rearranging the equation (6), it gets the equation for the error:

$$\text{Error} = \Delta - D_{T4}^{A\to B} = \frac{S^{uc}}{2} + \frac{R^{uc}}{2} + \frac{P^{uc}}{2} + \frac{RD_{T1\to T4}^{A\to B}}{2} \quad (10)$$

Synchronization between two devices

Equation (10) shows us how the error is composited. The uncertainty at the sender, uncertainty at the receiver, uncertainty in propagation time and the drift among the local clock together contributes the error. However, as mentioned in the design section, the uncertainty of the sender (send + access +transmission time) is the deterministic factor and other factors take much less compared with it. Therefore, in the following, a testing is set up based on the data collection of the uncertainty of the sender.

The testing is set up on Android devices of Samsung I9505 and Samsung Note2. I measure the transmission time at both devices 10 rounds and use Figure 2 to plot the uncertainty of the sender



The difference in transmission time S^{uc} contributes $\frac{S^{uc}}{2}$ time units to the total error. The average magnitude is around 3.07 microseconds. This implies that on average, the uncertainty at sender contributes a synchronization error about 1.54 microseconds.

This value is small and acceptable and it means TPSN algorithm is already successfully deployed in the WiFi-Direct network.

Message complexity

Comparing with traditional NTP protocol, TPSN reduces the overhead on the server node. In traditional NTP protocol, a server node may be linked to several client nodes and is responsible to react to all requests from the clients. This causes the server node a much higher overhead on message transferring. This is not a problem in the PC environment, because the power is unlimited in the PC environment. However, embedded devices are usually power-limited, especially it consumes quite much power for message transferring. TPSN may reduce power consuming by balancing the overhead to higher level nodes. One node in the TPSN network is responsible to react the requests from only one level higher, i.e. level 0 node is responsible for react to level 1 nodes only without considering requests from level 2 nodes or higher.

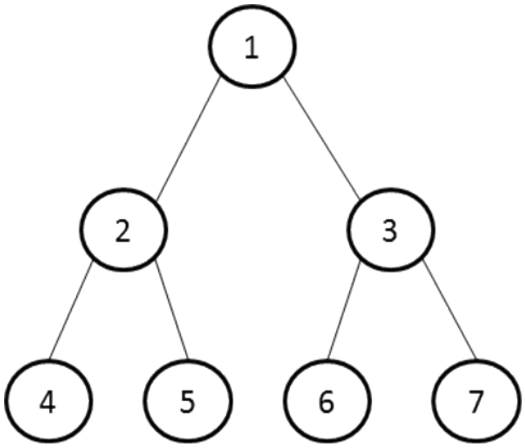


Figure 4: Connected Nodes

Considering a connected network as is connected in Figure 4, Node 1 has two child nodes of node 2 and 3. Each child nodes has another two child nodes, respectively. If we consider synchronizing the time clock in such a network using NTP protocol, level 1 nodes is responsible to react to all the requests and its message overhead is 4*(N-1) N equals the number of the nodes in the network. This number can be reduced to 4*2 for the TPSN network. Because in TPSN network, level 1 node reacts to its child nodes only and in Figure 4 for example, level 1 node reacts to node 2 and node 3 only. Requests from node 4,5,6,7 can be processed by nodes 2 and 3. When the network scales, newly joined nodes could be connected to higher level nodes without increasing overhead on the low level nodes.

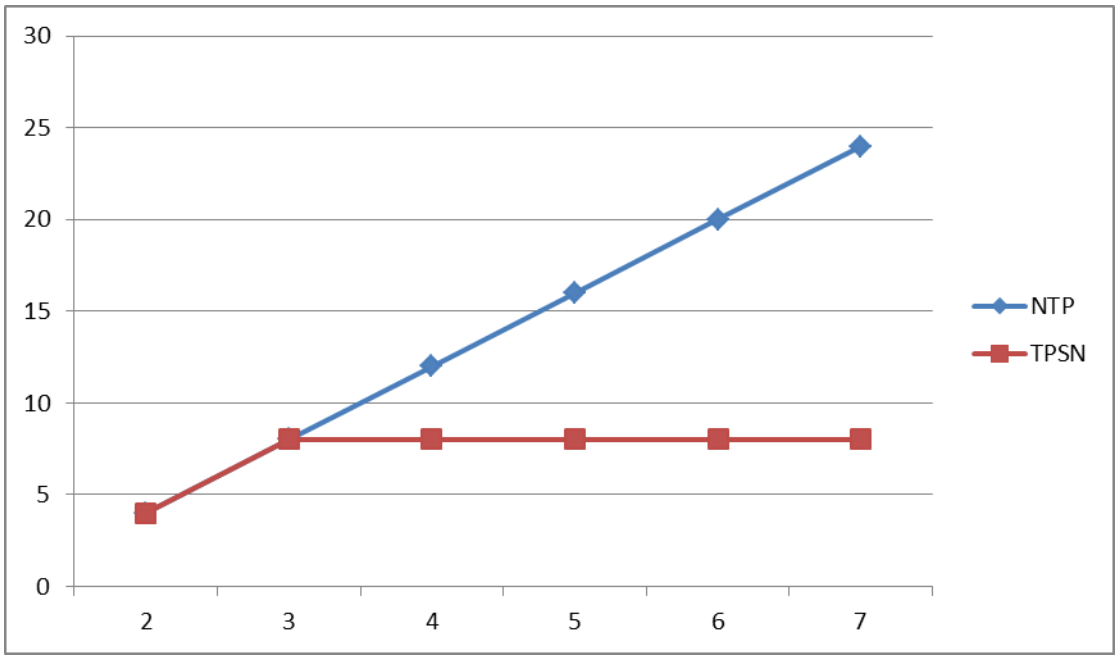


Figure 5: Level 0 node messages overhead

Figure 5 shows the message overhead for the root node when each node is allowed to have 2 child nodes. The horizontal axis is number of nodes while the veridical axis is the number of messages overhead. We can find that TPSN protocol may reduce the message complexity of the root nodes. In fact, the overhead for this part is delivered to its child of level 1 node.

6. Discussion and Conclusion

In this section, I will discuss further about this thesis and summary the findings and finally make a conclusion for this thesis.

Discussion

In this thesis, the problem of clock synchronization in WiFi-direct network is explored and researched. Android WiFi direct network replace Bluetooth or other type of local network connection and becomes very popular in the market. Clock synchronization in this network may be very useful and encourage developers to base the synchronized clock time to develop more applications. In this thesis, several algorithms are researched, it shows that traditional clock synchronization algorithm is not enough to solve this problem on the embedded devices; several algorithm for embedded devices with low overhead are raised to solve this problem. Timing Synchronization for Sensor Network (TPSN) algorithm is finally selected to create the

final solution. This algorithm applies a sender-receiver connection between nodes and deploys a simple algorithm to synchronize the time. In this algorithm, the synchronization error is mainly composited by three parts: uncertainty at the sender, uncertainty at the receiver and the uncertainty during the propagation time. In the data collection part, only the uncertainty at the sender is collected, this is because the uncertainty of the sender is the deterministic factor for the synchronization error and the other two parts are very small compared with the former factor. The collected data shows that the synchronization error is small and acceptable. TPSN is already successfully deployed to WiFi-Direct network. We can also find that TPSN has much better performance of message complexity when the network scales, it deliver the overhead to its child nodes without increasing much overhead on the root node. This allows the root node less power consumed and also helps the network scales.
Till now the presented work still remains in single-hop network. However, TPSN allows more devices to join the network and this is why it is attractive. In a multi-hop network, there will be different level nodes and these different level nodes can relay the synchronization work between the low level nodes and high level nodes. This topology allows each device responsible for only several nodes instead of too many. The synchronization work is very power-consuming therefore it requires more nodes to balance the work. This part of feature is already implemented in the implementation section. Another issue is about topology changing. When a node enters or quits the network, it may change the network topology. Especially, if a middle level node quits the network, all the higher nodes connects to this node should be able to connect to other nodes nearby or regenerate a new same level node. This feature is not considered well in this thesis and may be it can be implemented in the future.

Conclusion

WiFi-direct network is a new kind of network connection and it may attracts more attention in the future. In this thesis, it introduces how to synchronize the clock time in WiFi-direct network. It already shows that why traditional algorithm is not enough to solve the clock synchronization problem is WiFi-direct network. Several possible solutions are discussed here and TPSN is finally selected to implemented in my work. I have already showed how to use WiFiP2PManage to implement this algorithm synchronize clock time in Android WiFi-Direct network. It shows the results are good and acceptable. In the future, it should consider more about the multi-hop connection and considering the topology changing. It can conclude that TPSN is TPSN is a simple and efficient algorithm and be used to solve the clock synchronization problem in WiFi-direct network.

Acknowledgement

The author would like to express the gratitude to all teachers in Software Engineering and Management and who helped the author during the writing of this thesis. Many gratefully acknowledge the help of my supervisor, Lars Pareto and Morgan Ericsson, spent much time reading through each draft and provided me with inspiring advice. Also gratefully acknowledge the help of my team leader Zhu Rong, guiding my thesis as well as give me related technical support.

Reference

[1] David L. MillsRequest (1992), "RFC1305 - Network Time Protocol (Version 3) " University of DelawareObsoletes RFC-1119 , RFC-1059 , RFC-958

[2] Kay Römer (2001), "Time Synchronization in Ad Hoc Networks", IN ACM SYMPOSIUM ON MOBILE AD HOC NETWORKING AND COMPUTING (MOBIHOC 01

[3] J. Elson, L. Girod, and D. Estrin (2002), "Fine-grained network time synchronization using reference braod-casts", Technical Report UCLA-CS-020008, Uni-versity of California, Los Angeles

[4] Saurabh Ganeriwal, Ram Kumar, Mani B. Srivasrtava, "Timing sync Protocol for Sensor Networks", Network and Embedded Systems Lab (NESL), University of California Los Angeles 56-125B Eng. IV, UCLA EE Dept., Los Angeles, CA90095

[5] D. L. Mills, “Internet time synchronization: The Network Time Protocol” In Z. Yang and T.A. Marsland, editors, Global States and Time in Distributed System s. IEEE Computer Society Press, 1994

[6] D. Mills Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, rfc 4330, 2010

[7] Android SDK, [online available http://developer.android.com/guide/topics/wireless/wifip2p.html], 2012

[8] L. Pelusi, A. Passarella, and M. Conti, “Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks,” IEEE Communications Magazine Decenber 2006.